LDA script Josh

Joost Bloos

07/11/2021

```
#Load packages and libraries required
#install.packages("dplyr")
#install.packages("tidyr")
#install.packages("quanteda")
#install.packages("lda")
#install.packages("LDAvis")
#install.packages("servr")
#install.packages("tm")
```

```
library(LDAvis)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(quanteda)
```

```
## Package version: 3.1.0
## Unicode version: 13.0
## ICU version: 69.1
```

```
## Parallel computing: 4 of 4 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
library(stringr)
library(lda)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'

## The following objects are masked from 'package:quanteda':
##
##     meta, meta<-

##
## Attaching package: 'tm'

## The following object is masked from 'package:quanteda':
##
##     stopwords
```

```
#read data set Tweets May 16, 2020: Covid related hastags as per project document.
#getwd()
#setwd("C:/Ryerson University - Capstone project/Module 2/EIEEE - Large dataset/Combined")
data_set_may <- read.csv("corona_tweets_59 May 2020", header = T, sep = ",")
```

```
## Warning in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, :
## embedded nul(s) found in input
```

```
#take a sample of 1,000, set seed to replicate results across several analysis of methods:
set.seed(1000)
rawData <- data_set_may[sample(nrow(data_set_may), size = 1000), ]
#str(rawData)
```

```
#create a corpus:
importdocs = corpus(rawData, text_field = 'text')
```

```
#code remove digit creates an error on JSON object as not all characters are zero...
importdocs = gsub("[[:digit:]]", " ", importdocs) #remove digits Covid-19 = Covid19 = Covid
```

```
#preprocessing of data (large original dataset)
importdocs <- gsub("'", "", importdocs)  # remove apostrophes
importdocs <- gsub("[[:punct:]]", " ", importdocs)  # replace punctuation with space
importdocs <- gsub("[[:cntrl:]]", " ", importdocs)  # replace control characters with space
importdocs <- gsub("^[[:space:]]+", "", importdocs) # remove whitespace at beginning of documents


#importdocs <- str_replace_all(string=importdocs, pattern= "[-ïâžÿŸ#œ‰â???¦TðY¥]" , replacement= "") #r


importdocs <- str_replace_all(string=importdocs, pattern= "https" , replacement= "")
importdocs <- str_replace_all(string=importdocs, pattern= "amp" , replacement= "")

importdocs <- tolower(importdocs)



importdocs <- gsub("[[:space:]]+$", "", importdocs) # remove whitespace at end of documents
```

```
doc.list <- strsplit(importdocs, "[[:space:]]+")



length(doc.list)
```

```
## [1] 1000
```

```
# compute the table of terms:
term.table <- table(unlist(doc.list))
term.table <- sort(term.table, decreasing = TRUE)
head(term.table)
```

```
##
##   the     t    co    to    of covid
##  1035   901   888   789   548   506
```

```
# remove terms that are stop words or occur fewer than 5 times:
#stop_words <- stopwords("SMART") replaced after throwing an error non-zero characters when creating JS
stop_words <- stopwords("SMART")

del <- names(term.table) %in% stop_words | term.table < 5
term.table <- term.table[!del]
vocab <- names(term.table)
head(vocab, 10)
```

```
##  [1] "covid"       "coronavirus" "deaths"      "people"      "cases"
##  [6] "trump"       "pandemic"    "world"       "corona"      "health"
```

```
# now put the documents into the format required by the lda package:
get.terms <- function(x) {
  index <- match(x, vocab)
  index <- index[!is.na(index)]
  rbind(as.integer(index - 1), as.integer(rep(1, length(index))))
}
documents <- lapply(doc.list, get.terms)
#documents
```

```
# Compute statistics related to the data set:
D <- length(documents)  # number of documents (1,000)
W <- length(vocab)  # number of terms in the vocab (567)
doc.length <- sapply(documents, function(x) sum(x[2, ]))  # number of tokens per document [6, 4, 13, 4,
N <- sum(doc.length)  # total number of tokens in the data (6,764)
term.frequency <- as.integer(term.table)

D
```

```
## [1] 1000
```

```
W
```

```
## [1] 570
```

```
head(doc.length)
```

```
## [1]  6  4 13  4  5  9
```

```
N
```

```
## [1] 6704
```

```
# MCMC (Markov Chain Monte Carlo - gibbs sampling) and model tuning parameters:
K <- 6   ## this is the number of topics.
G <- 5000
alpha <- 0.02
eta <- 0.02
```

```
# Fit the model:
# install.packages("lda")
# library(lda)
set.seed(357)
t1 <- Sys.time()
fit <- lda.collapsed.gibbs.sampler(documents = documents, K = K, vocab = vocab,
                                   num.iterations = G, alpha = alpha,
                                   eta = eta, initial = NULL, burnin = 0,
                                   compute.log.likelihood = TRUE)
t2 <- Sys.time()
t2 - t1  # about 20 seconds on laptop
```

```
## Time difference of 14.43811 secs
```

```
#apply calculations:
phi <- t(apply(t(fit$topics) + eta, 2, function(x) x/sum(x)))
theta <- t(apply(fit$document_sums + alpha, 2, function(x) x/sum(x)))
```

```
#combined into one dataset:
#replace object to tweetsList

tweetsList <- list(phi = phi,
                     theta = theta,
                     doc.length = doc.length,
                     vocab = vocab,
                     term.frequency = term.frequency)
```

```
#unsuccessful to knit the following LDA presentation, code should work!

#create the JSON object to feed visualization:
json <- createJSON(phi = tweetsList$phi,
                   theta = tweetsList$theta,
```

```
                  doc.length = tweetsList$doc.length,
                  vocab = tweetsList$vocab,
                  term.frequency = tweetsList$term.frequency)


#unsuccessful to knit the following LDA presentation, code should work!
#serVis(json, out.dir = 'vis', open.browser = TRUE)
```

```
#after discussing which script to use due to issue with the Lab modul 10 script, add heatmap. For now t
```

```
#install.packages("caret", dependencies = c("Depends", "Suggests"))
```